Telecommun Syst (2013) 52:2163–2176
DOI 10.1007/s11235-011-9539-8

Preserving privacy against external and internal threats in WSN data aggregation

Lei Zhang · Honggang Zhang · Mauro Conti · Roberto Di Pietro · Sushil Jajodia · Luigi Vincenzo Mancini

Published online: 2 August 2011

Abstract In this paper, we propose two efficient and privacy-preserving data aggregation protocols for WSNs: PASKOS (Privacy preserving based on Anonymously Shared Keys and Omniscient Sink) and PASKIS (Privacy preserving based on Anonymously Shared Keys and Ignorant Sink)—requiring low overhead. Both protocols guarantee privacy preservation and a high *data-loss resilience*. In particular, PASKOS effectively protects the privacy of any node against other nodes, by requiring $O(\log N)$ communication cost in the worst case and $O(1)$ on average, and $O(1)$ as for memory and computation. PASKIS can even protect a node's privacy against a compromised sink, requiring only $O(1)$ overhead as for computation, communication, and memory; however, these gains in efficiency are traded-off with a (slightly) decrease in the assured level of privacy.

A thorough analysis and extensive simulations demonstrate the superior performance of our protocols against existing solutions in terms of privacy-preserving effectiveness, efficiency, and accuracy of computed aggregation.

Keywords Wireless sensor network security · Data aggregation · Hierarchical aggregation · Attack-resilient · Privacy

L. Zhang · M. Conti · S. Jajodia
George Mason University, Fairfax, VA, USA

L. Zhang
e-mail: lzhang8@gmu.edu

S. Jajodia
e-mail: jajodia@gmu.edu

H. Zhang
Suffolk University, Boston, MA, USA
e-mail: honggang@cs.umass.edu

M. Conti (✉)
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
e-mail: mconti@few.vu.nl

R. Di Pietro
Università di Roma Tre, Rome, Italy
e-mail: dipietro@mat.uniroma3.it

L.V. Mancini
Università di Roma La Sapienza, Rome, Italy
e-mail: mancini@di.uniroma1.it

1 Introduction

Applications of Wireless Sensor Networks (WSNs) range from military surveillance to environmental monitoring. WSNs applications, such as domotics and health care [1, 2], have recently also been envisioned to help our daily lives. An important issue with the latter applications is the *privacy* of collected information. For instance, people might not want to disclose their personal information when WSNs are used to measure power or water usage of their houses (this information can be useful to the provider of the commodity [1] for optimizing the distribution of resource). The privacy issue is even more compelling and evident in health care applications [2]. Privacy is also important in surveillance applications; for instance, data need to be protected in a WSN collecting personally identifiable information such as sensing the identities of people in a building as part of an access control system [3].

In this paper, we address the challenge of designing efficient *privacy-preserving* protocols for data aggregation in wireless sensor networks. Intuitively, the data aggregation should be executed in a way that, besides the aggregation result, no information from a single node can be revealed to any other entities.

In a typical WSN, a number of sensor nodes are deployed in an area of interest and they self-organize to form a network. The values sensed by sensor nodes are collected by a *sink* or *base station* (which is computationally more powerful and secure than sensor nodes) for further processing. For typical applications where the sink is only interested in some kind of aggregate value such as the average of all sensed values, the aggregate value is collected via an *in-network* data aggregation process (for reducing energy consumption) in which partial aggregate values are combined at intermediate nodes en route to the sink. This process is realized by constructing a spanning tree among sensor nodes with the sink directly connected to the root sensor node, and each sensor node aggregates its incoming data with its sensed value (privately owned), and then sends the result to its parent node in the tree [4, 5].

Note that for the data aggregation process in its basic form, the private sensed value of a sensor node can be easily inferred by a curious neighbor node (within wireless communication range) or an eavesdropper. In this paper, to protect the privacy of a sensor's sensed value we strive to design a protocol for the aggregation process with the following security and operating requirements: (i) *privacy-preserving*. The protocol can protect the private sensed value of a node from being disclosed to any other sensor node, the sink, an adversary that compromises a portion of the network nodes, or even an adversary that compromises the sink and can eavesdrop all network traffic; (ii) the protocol is *data-loss resilient* [6] in the sense that it will always compute an intended aggregation function (e.g., average) over the actual contributing data values, despite of any message loss or node failure; (iii) the protocol is efficient in terms of computation and communication so to reduce energy consumption and increasing the life time of a WSN. To the best of our knowledge, the existing privacy-preserving data aggregation algorithms in [1, 6–8] do not completely satisfy these requirements. We should notify that among the above privacy-preserving requirements, to protect the single node's privacy against the *sink* is also important. For example, in a health care application [2] where drug supplement is investigated, privacy is of high value. Furthermore, the privacy-preserving algorithms for data mining applications [9, 10] are computationally too expensive for the resource constrained sensor nodes.

To meet the above requirements, we propose two protocols: PASKOS (Privacy-preserving based on Anonymously Shared Keys and Omniscient Sink) and PASKIS (Privacy-preserving based on Anonymously Shared Keys and Ignorant Sink). In both protocols, each sensor node is set up to anonymously share a number of secret keys with other nodes or the sink. In PASKOS, the sink is omniscient, i.e., it possesses every secret key shared by the sensor nodes; in PASKIS, the sink is oblivious, i.e., it does not know any of the shared keys. In both protocols, each node will compute a secret *keyed value* using the shared secret keys and submit the sum of its sensed value and the *keyed value* to the aggregation process. In PASKOS, the sink will retrieve the true aggregate of sensed value by removing the keyed values himself. In PASKIS, the sink will directly obtain the true aggregate result because the keyed values are removed by nodes themselves during the aggregation process. The effective privacy-preservation of each node's sensed value is due to the small probability for any two nodes to share a relatively large number of keys.

Our protocols are inspired by the technique in [6], in which during the aggregation phase, each node submits to the aggregation process the sum of its sensed value and a secret value shared between this node and the sink, and the sink can recover the final aggregate of sensed values by removing all the secret values from its received data. The technique in [6] ensures that the privacy of each node's sensed value is protected by the secret values, but it does not provide privacy protection of a node against the sink that can eavesdrop the node's incoming and outgoing traffic, and that the technique in [6] is not data-loss resilient, i.e., a single message loss will cause the sink cannot correctly remove all keyed values. Compared to [6], both our protocols are data-loss resilient, and PASKIS can even protect a node's privacy from the sink. In addition, note that both our protocols' key assignment process pre-assign to each sensor node a set of keys randomly chosen from a large key pool, similarly to [11]. However, the scheme in [11]—superseded in [12]—attempts to ensure each pair of neighbor nodes share keys and more shared keys implies a better secure communication service. Whereas, our protocols take advantage of the pre-assigned keys in a different way: any two neighbor nodes are very unlikely (with very low probability) to be assigned with the same set of keys.

Our contributions.

- We propose the PASKOS protocol that can protect the privacy of a node from other sensor nodes, and it also achieves a high privacy-preserving assurance when a portion of network nodes is compromised (but not the sink) by an adversary who can eavesdrop all network traffic. The computation and memory overheads in each node are $O(1)$ in the worst case; the communication overhead in each node is $O(1)$ on average, and $O(\log N)$ in the worst case.
- We propose the PASKIS protocol that can protect the privacy of a sensor node even when the sink and a portion of network nodes are compromised at the same time, besides the privacy-protection provided by PASKOS. The computation, memory and communication overheads in each node are all $O(1)$ in the worst case. Compared with the PASKOS protocol, the higher efficiency of the PASKIS

protocol comes with a slightly higher probability of privacy violation when a portion of sensor nodes (but not the sink) are compromised by an adversary.
– Through formal analysis and simulations, we demonstrate our protocols' privacy-preserving effectiveness, efficiency, and data-loss resilience.

We finally remark that the aim of our work is to guarantee the privacy of the nodes participating in the aggregation mechanism. The security of the aggregation mechanism itself is out of the scope of the paper. In particular, the proposed solution is vulnerable to Denial of Service attack. However, other solutions exist in the literature to cope with this issue. For instance, other works address the problem of checking if the collected aggregate has been compromised [13–17] or even identify the malicious node [17]. Note that, this problem is also closely related to the problem of "Outlier Detection" [18].

Roadmap. The rest of the paper is organized as follows. In Sect. 2 we present the related work. Section 3 presents the assumption and the notation used throughout the paper. In Sect. 4 we propose our PASKOS protocol, while Sect. 5 details the PASKIS protocol. The analysis and simulation-based evaluation of the proposals can be found in Sect. 6. Concluding remarks are reported in Sect. 7.

2 Related work

The need for saving energy in WSNs has urged researchers to introduce the in-network aggregation paradigma. The first proposals, such as [4, 5], were designed to compute aggregates such as Count, Sum, and Average. However, security and privacy issues were simply ignored. As for security, a few interesting proposals have been enriching the literature corpus, such as [13, 15, 16, 19, 20]; however, none of the above algorithms was designed to address privacy issues.

It is interesting to note that privacy issues in computing aggregate has already been addressed in both statistical database and data mining applications. For instance, in [9, 10], the authors proposed data perturbation techniques to protect values privacy, whereas a few secure multi-party computation schemes were designed in [21–23]. However, these solutions require data to be centralized in order to process them. Further, these privacy-preserving algorithms generate a heavy computing load; something that does not fit the resource constrained capabilities in a WSN. Finally, even if some recent proposals [24, 25] considerably reduces the computational overhead, WSNs still require ad-hoc solutions to fit their hardware constraints. An interesting proposal appeared in [3] introduces the possibility to select the appropriate privacy preserving policy, when querying a WSN. There, different techniques supporting the described goal are detailed. A recent privacy issue in WSNs relates to the privacy of the ID of nodes that signal an event. Problem statement and preliminary results are provided in [26]. The same problem is addressed in [27], highlighting a technique focusing on preserving the anonymity of the sensor ID that provides the data of interest, rather than on the privacy of the data itself. However, the protocol provided relies on onion-routing like solution, hence introducing a non-negligible burden on both computational and communication. Further, the level of privacy achieved still needs to be fully validated.

The foundations of math that allow to perform computations over encrypted data are referred in computer science as Privacy Homomorphism (PH). The first Privacy Homomorphism (PH) was proposed by Rivest et al. [28] to allow aggregation of encrypted data. In Girao et al.'s work [7], a PH construction is used to allow the aggregator node to fuse the locally sensed data with the encrypted aggregate received from contributing sensor nodes. However, the protocol does not guarantee the privacy of individual sensed data either against other nodes or against the sink.

In the following we report the state of the art as for privacy preserving data aggregation protocols; we will compare our proposals against the most representative protocols, as summarized in Table 1.

In [6], the authors propose a solution for data aggregation that preserves the privacy against other nodes. The authors assume that each node n_i shares a key k_i with the sink and always adds a secret number as noise, determined by k_i, to its sensed value. The BS can remove, from the received aggregate, all the noises because he shares the keys with all nodes. We observe that this scheme does not protect privacy of individual sensed values against the sink and, moreover, is severely vulnerable to message loss, which is common in a WSN. In particular, the sink will obtain a bogus aggregate even with a single message loss. The authors propose to cope with this last issue by adding either the list of contributing nodes or the list of nodes that did not contribute (whichever is shorter).

However, this solution does not prevent this list from being $O(N)$ in length—in the worst case—, where N is the number of sensors in the network. In [1], the authors propose two different solutions: CPDA and SMART. The former gives a solution for data aggregation preserving node-privacy against other nodes. However, as the same authors recognize, the overhead of this protocol is high. The latter proposal, SMART, is more efficient than the first one. However, it suffers from the same problem of message-loss as in [6]. Furthermore, in [29] the authors addressed the problem, by which the solution in [6] is affected, i.e., the sink has to know the list of the nodes participating in the aggregation. However, these solution come at the cost of an increase in the computational complexity.

Another recent work addressing privacy is [8]; however the solution proposed is not resilient against a compromised

Table 1 Private Data Aggregation protocols: Comparing the security features and complexities. †: bounded by a pre-determined constant parameter K for the protocol. ‡: bounded by a pre-determined constant parameter P for the protocol

	Aggregation type	Encryption type	Privacy from eaves.	Privacy from other nodes	Privacy from the sink	Data-loss resilience	Node comput. complexity	Commun. complexity
CDA protocol [7]	Hop-by-hop	CH-to-BS	Yes	No	No	**Yes**	$O(1)$	$O(1)$
Protocol in [6]	Hop-by-hop	Node-to-BS	Yes	**Yes**	No	No	$O(1)$	$O(1)$
CPDA protocol [1]	CH	Node-to-CH	Yes	**Yes**	**Yes**	**Yes**	$O(C^2 \log C)$	$O(C)$
SMART protocol [1]	Hop-by-hop	Node-to-node	Yes	**Yes**	**Yes**	No	$O(1)$	$O(C)$
O-ASP [29]	Hop-by-hop	Node-to-BS	Yes	**Yes**	No	**Yes**	$O(1)$	$O(C)$
D-ASP [29]	Hop-by-hop	Node-to-BS	Yes	**Yes**	No	**Yes**	$O(1)$	$O(C)$
Mlaih et al. [8]	Hop-by-hop	Node-to-BS	Yes	**Yes**	No	No	$O(1)$	$O(1)$
PASKOS—Sect. 4	Hop-by-hop	Node-to-node	Yes	**Yes**	No	**Yes**	$O(1)^{\dagger}$	$O(\log N)^{\ddagger}$
PASKIS—Sect. 5	Hop-by-hop	Node-to-node	Yes	**Yes**	**Yes**	**Yes**	$O(1)^{\dagger}$	$O(1)^{\ddagger}$

sink. Further, tight message synchronization constraints are placed on both the sink and the nodes in the network. Indeed, if an aggregated message sent by a node to the sink does not reach the sink, any further aggregation phase will fail. Similar message synchronization problems have been pointed out also when dealing with privacy for RFID systems [30].

We recently proposed PASKOS and PASKIS [31]. PASKOS effectively protects the privacy of any node against other nodes, by requiring $O(\log N)$ communication cost in the worst case and $O(1)$ on average, and requiring $O(1)$ memory and computation cost. PASKIS can even protect a node's privacy against a compromised sink, and it is more efficient, requiring only $O(1)$ overhead as for computation, communication, and memory; however, these gains in efficiency are traded-off with a (slightly) decreased level of privacy. The present work is an extension of [31]. In the current version of the paper, we give a more detailed description and performances investigation of the protocols proposed in [31].

In Table 1 we summarize the comparisons of our protocols with other existing privacy-preserving protocols/algorithms. *Hop-by-hop* aggregation type means that each node adds its own value to the aggregate; *CH* aggregation type means that a local aggregation is performed by the cluster head (one selected node in the cluster). The *Encryption type* column refers to the nodes that are involved in an encryption and decryption process. In particular, *Node-to-BS* means that each node encrypts some data that cannot be decrypted until it reaches the BS. In the *Privacy from other nodes* column we assume some nodes are compromised by an adversary that can also eavesdrop all network traffic. In *Privacy from the sink* column we assume the sink and a portion of sensor nodes are compromised by an adversary that can also eavesdrop all network traffic. The column *Data-loss resilience* refers to the fact that the base station can compute the correct aggregate if a number of nodes do not participate in the protocol or if a message is lost. Finally, C and N represent the size of a cluster (as a privacy protection parameter) and the number of nodes respectively.

Note that except CPDA in [1], the other existing privacy-preserving protocols (including CDA [7], the protocol in [6], SMART [1], O-ASP and D-ASP [29], the protocol in [8]) cannot simultaneously provide data-loss resilience and satisfy all the privacy-preserving requirements mentioned above—with the exception of CPDA. However, when CPDA is compared to our proposal, it turns out that it is operationally much more complicated and has higher computational and communication complexity.

3 Preliminaries

3.1 Network model

We consider a wireless sensor network in which a sink or base station collects aggregated information of the sensed values of all nodes in the network. Typical aggregated information includes *sum, average, min, max*, and *count*. In this paper, we focus on the sum of sensed values of all sensor nodes. Finding many other aggregated information such as *average, count* can be reduced to finding *sum* [1, 6].

A typical data aggregation process [4] works as follows. Initially in *phase 1 (request broadcast phase)*, the sink broadcasts an aggregation request. Each node sets as its parent the node from which it hears the request and rebroadcasts the request. A logical aggregation tree is built in the network in this phase. In *phase 2 (aggregation phase)*, the leaf nodes of the logical tree will send their sensed values to their parent nodes. Each non-leaf node aggregates all received values (from its children) with its own sensed value,

and then sends its aggregate to its parent node. In the end, the sink receives the aggregate of the sensed value of all nodes that have participated in the aggregation process. Note that each node only needs to communicate with its neighbor nodes in the aggregation tree.

3.2 Criteria for efficient privacy-preserving data aggregation protocol

Effectiveness of privacy-preserving *First*, we require that the private sensed value of any node should not be disclosed to any other node in the network or even the sink. *Second*, in the case that a powerful adversary is able to eavesdrop all network traffic and compromise a portion of nodes or even the sink, a good protocol should be able to reduce as much as possible the probability that the private sensed value of an un-compromised node is disclosed to the adversary.

Efficiency Recall that in-network processing schemes in a WSN are developed to reduce the power and bandwidth usage for resource constrained WSNs. Thus, a good protocol should not impose a large amount of overhead on communication, computation, and memory, for the sake of achieving any privacy-preserving goal.

Data loss resilience In WSNs, message loss essentially drops some nodes from an aggregation. Thus after removing all privacy-preserving related information, the sink should be able to correctly retrieve the aggregate of the sensed values of only those nodes who actually have participated in the aggregation. This property is referred to as *data-loss resilience*. For instance, the scheme in [6] is not data-loss resilient if the sink does not know the list of actual participating nodes.

3.3 Solution overview

Our goal is to design privacy-preserving data aggregation protocols that achieve the aforementioned effectiveness, efficiency, and data loss resilience. To this end, we propose two protocols: PASKOS and PASKIS. The key idea of our protocols is outlined below.

We introduce *keyed values* into a typical data aggregation protocol such as TAG [4]: each node computes a *randomized value* by adding a secret *keyed value* to its sensed value and submits only the *randomized value* during the data aggregation. The *keyed value* is computed (through a hash function) from some secret keys held by the node. More specifically, assuming each sensed value is an integer, the computations for the *randomized value* and the data aggregation are done using modular arithmetic, with the same modulus M. For example, the *randomized value* is computed as follows:

$$\text{randomized value} \equiv \text{sensed\_value} + \text{keyed\_value} \; mod \; M$$

We select for M a value that is sufficiently large, i.e., $M > N \cdot s_d$, where N is the number of nodes in the network and s_d is the upper bound for the sensed value (assumed to be non-negative). This is a necessary condition for the sink to remove all *keyed values* from the aggregation result to obtain the precise sum of the sensed value. Note that we assume the *keyed value* is expressed on a number of bits that is enough to protect the sensed value. For the sake of simplicity, we will use "+" and "−" as modular addition and subtraction in the remainder of this paper.

We recall that in [6] each node shares a secret key with the sink and uses it to compute the *keyed value*. Thus, the sink can compute all the keyed values by itself and remove them from the aggregation result. Again, it is worth noting that this solution: (i) is not data-loss resilient; (ii) cannot protect a node's privacy against the sink, if the sink can eavesdrop a node's communication content.

Compared to [6], our protocols leverage the idea of "anonymous sharing". Specifically, each node, instead of sharing a secret key with the sink, is pre-assigned with a set of keys (i.e., a key ring) that are randomly chosen from a secret key pool (similar to Eschenauer and Gligor's scheme [11]). Each node knows the IDs of all the keys in the key pool, and each node keeps its own key ring secret to itself. Thus each node shares its keys anonymously with other nodes, and the probability that any pair of nodes share the same key ring is very small. The lower this probability, the higher the privacy provided by our protocols. In PASKOS, we let the sink possess the whole key pool, whereas in PASKIS, the sink does not possess any key.

These "anonymously" shared keys will be used to compute the *keyed values* by each node. Note that in [11], the pre-assigned keys are used to establish unique session keys between any two nodes so that the selection of key pool size and the key ring size should be dependent on network size, wireless communication range, etc. However, the key-assignment process for our protocols has completely different purpose as mentioned above. For preserving the privacy of each node, we utilize the fact that the probability for any two nodes to share the same set of pre-assigned keys are very low. Thus, we can select the key pool size P and the key ring size K as pre-determined constant for the privacy protection. Note that in PASKIS, we need to make sure any key that is used in aggregation process should be at least shared (anonymously of course) by a pair of nodes, which can be guaranteed by the operation of our protocols.

Table 2 lists the notations used in our protocols.

4 The PASKOS protocol

In this section, we present the PASKOS protocol, which can protect the privacy of a node against any other sensor nodes

Table 2 Notations used in our protocols

N	Number of nodes in the network
$n_0 \ldots n_{N-1}$	Nodes' IDs
P	Key-pool size, number of different keys in total
$\mathcal{K}^j$	Key ring of node n_j, containing assigned keys
K	Key-ring size, number of keys assigned to each node
k_i	i-th key in the overall key pool
$Seed$	Identification number of an aggregation process
BIT_Q^j	P-bit key bitmaps of node n_j (request broadcast phase of PASKIS protocol)
BIT_A^j	P-bit key bitmaps of node n_j (aggregation phase of PASKIS protocol)
$bit_{i,Q}^j$, $bit_{i,A}^j$	i-th bit of BIT_Q^j and BIT_A^j, respectively. PASKIS protocol only
d_j	Value of sensed value of node n_j, assumed to be integers
D_j	Aggregated value of node n_j
$Hash$	A secure hash function used by all nodes
$+/-$	$+/- \ mod\ M$, where M is chosen to be larger than $N \cdot s_d$, and s_d is the upper bound (assumed to be non-negative) for sensed values

in a WSN. The basic idea of this protocol is that each node uses all of its keys in its key ring to compute its *keyed value*, and the sink is omniscient (i.e., possesses the whole key pool).

Before describing the PASKOS protocol, we observe that this differs from solutions where each node shares a different symmetric key with the sink (such as in [6]) and each node uses that key to make private its own aggregated value. In particular, the differences are twofold:

- first, the current solution (i.e. [6]) does not have mechanisms for data loss resiliency—if just one node is not able to participate in the aggregation, the sink cannot retrieve the correct aggregate.
- a fix to the previous problem can be made for [6] adding to the forwarded aggregated value the information about the ids of the nodes that actually participated in the aggregation. However, this information would have a $O(N)$ message size overhead, rather than $O(P)$ of our proposal. As P (the key pool size we consider) is independent from N (the size of the network) this make our proposal based on random keys assignment more scalable than just extending previous solutions.

4.1 Protocol description

In the request broadcast phase, the sink first sends its request to one node, denoted as n_0. Then starting from n_0, the request is broadcast to the entire network in a hop-by-hop fashion. As a result, an aggregation tree (rooted at n_0) is formed. In the aggregation phase, typically the sensed values of all sensor nodes get aggregated as they propagate from leaf nodes up to the root node n_0. With the PASKOS protocol, nodes perform the following operations.

Procedure 1 *PASKOS_Leaf_Aggregate*

1: /*Node n_a executes this procedure*/
2: $D_a = d_a$ /*d_a is the sensed value of node n_a*/;
3: **for** each $k_i \in \mathcal{K}^a$ **do**
4: $\quad c_i^a =$ a randomly picked value from $\{1, -1\}$;
5: $\quad D_a = D_a + c_i^a \cdot Hash(Seed, k_i)$;
6: **end for**
7: Node n_a sends $\langle D_a, C^a \rangle$ to its $Parent(n_a)$;

Procedure 2 *PASKOS_Non-Leaf_Aggregate*

1: /* Node n_a executes this procedure */
2: **for** each n_j in $Children(n_a)$ **do**
3: $\quad$ Receive $\langle D_j, C^j \rangle$ from n_j;
4: **end for**
5: $D_a = \sum_{n_j \in Children(n_a)} D_j$;
6: $D_a = D_a + d_a$;/*d_a is the sensed value of n_a*/
7: **for** each $k_i (1 \le i \le P)$ in the key pool **do**
8: $\quad temp = \sum_{n_j \in Children(n_a)} c_i^j$;
9: $\quad$ **if** $k_i \in \mathcal{K}^a$ **then**
10: $\quad\quad$ **if** $temp = 1$ or $temp = -1$ **then**
11: $\quad\quad\quad c_i^a = -temp$;
12: $\quad\quad$ **else**
13: $\quad\quad\quad$ Randomly select c_i^a from $\{1, -1\}$;
14: $\quad\quad$ **end if**
15: $\quad\quad D_a = D_a + (c_i^a - temp) \cdot Hash(Seed, k_i)$;
16: $\quad$ **else**
17: $\quad\quad c_i^a = temp$;
18: $\quad$ **end if**
19: **end for**
20: **if** $n_a \neq n_0$ **then**
21: $\quad$ Send $\langle D_a, C^a \rangle$ to its $Parent(n_a)$;
22: **else**
23: $\quad$ Send $\langle D_0, C^0 \rangle$ to the Sink;
24: **end if**

A leaf node n_a follows the sequence of steps shown in Procedure 1. It first computes a *keyed value*, $Hash(Seed, k_i)$, for each key k_i that it possesses. Note that *Seed* is the identification number of an aggregation process. Then, this node randomly chooses to add all keyed values to or subtracts them from this node's sensed value. This step is shown in line 5, where c_i^a (that takes on either the value 1 or -1) indicates the random choice (adding or subtracting, respectively). The resulting *randomized value* is denoted by D_a. We use C^a to denote the set or the vector of all coefficients used by node n_a.

Each non-leaf node n_a follows the algorithm described in Procedure 2. First, n_a aggregates the values received from its children with its own sensed value d_a (line 5 and line 6) to get D_a. Then, n_a computes a keyed value based on each pos-

sessed key k_i. Further, n_a adds or subtracts these keyed values from D_a in the following way. For each key k_i possessed by n_a (i.e., $k_i \in \mathcal{K}^a$), if the sum of coefficients used by the children of n_a is either 1 or -1 (line 10), then n_a sets c_i^a either to -1 or 1, respectively (line 11). Otherwise, n_a sets c_i^a to a randomly chosen value from set $\{1, -1\}$ (line 13). For each key $k_i \notin \mathcal{K}^a$ but used by its descendants, node n_a will simply set c_i^a to its received sum of coefficients. Note that lines 10 and 11 guarantee that any key k_i possessed by n_a is used in computing its D_a. Without these two lines, it would be possible that k_i is not used by n_a. For example, this could occur if $temp = 1$ and the c_i^a is set to 1 (with probability 0.5 based on line 13).

The sink—that holds the entire key pool—retrieves the sum of all sensed values D ($D = \sum_{j=1}^{N} d_j$) from its received final aggregated *randomized value* D_0 as follows:

$$D = D_0 - \sum_{i=1}^{P} c_i^0 \cdot Hash(Seed, k_i) \quad (1)$$

4.2 An example

We now show how the PASKOS protocol works through an example with 6 sensor nodes as shown in Fig. 1. Suppose that the key pool size is 4. In the request broadcast phase, an aggregation tree (rooted at n_0) is formed in this WSN. In the aggregation phase, leaf node n_5 computes its randomized value D_5 (based on Procedure 1), and then sends it together with $C^5 : (0, 0, 1, -1)$ (vector of coefficients (c_1, c_2, c_3, c_4)) to its parent node n_2. Note that in Fig. 1 we use S_i to denote the keyed value computed from key k_i and *Seed* (unique to each round of data aggregation); that is, $S_i = Hash(Seed, k_i)$. Similarly leaf nodes n_3 and n_4 send their D_3 and D_4 along with their coefficients to their parent node n_1. Then, following the algorithm in Procedure 2, non-leaf nodes n_1 and n_2 send their aggregate results and coefficients to their parent node n_0. Root node n_0 then sends the final aggregate D_0 along with its vector of coefficients to the sink.

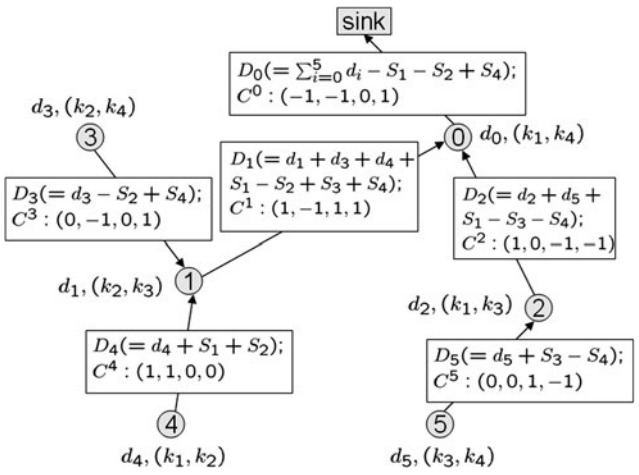


Fig. 1 An instance of the data aggregation process with the PASKOS protocol and an example WSN consisting of 6 sensor nodes and a sink. Keyed values: $S_i = Hash(Seed, k_i)$. Key pool contains 4 keys. Each transferred message consists of an aggregated randomized value and a set of coefficients

4.3 Analysis of PASKOS protocol

Privacy *First*, PASKOS protocol protects the privacy of any sensor node against any other node in the network. This is because, by choosing appropriate parameters K and P, our random key pre-assignment process guarantees that the probability of any two nodes sharing the same key ring is negligible. For instance, in the example shown in Fig. 1, node n_1 cannot infer its child's private sensed value d_3, based on its knowledge of k_2, k_3, D_3 and C^3. *Second*, PASKOS protocol protects a node's private sensed value against the sink, since the sink does not know the key rings of individual nodes, although the sink knows the entire key pool. *Third*, PASKOS can keep a node's privacy-disclosure probability at a desired low level. This is possible even in face of a powerful adversary that can compromise a portion of the network nodes (but not the sink) to obtain their stored information (such as key rings) and also can eavesdrop all network traffic. A detailed analysis of PASKOS is provided in Sect. 6.

However, if the adversary compromises the sink and is able to eavesdrop all network traffic, then this protocol fails to protect any node's privacy.

Correctness The sink can always obtain the aggregate of all sensed values by removing all keyed values from the received aggregate result, because the sink knows the entire key pool and root n_0 sends the vector of coefficients for used keys to the sink.

Efficiency In the PASKOS protocol, each node stores a constant (K) number of pre-distributed keys and uses them to compute *keyed values* via a hash function. The memory and computational overhead per node is bounded by the constant K. Now consider the communication overhead per node. In the worst case (also expected to be very rare) during the aggregation phase, the largest number of times that any pre-assigned key is used to compute the keyed values is bounded by N, stored in $O(\log N)$ bits. As the key pool size P is also a pre-determined constant, the worst-case communication overhead is bounded by $O(\log N)$. However, on average the communication overhead is $O(1)$. This is because the coefficients of the pre-assigned keys to compute the keyed values are uniformly at random chosen from $\{-1, 1\}$.

5 The PASKIS protocol

The PASKOS protocol proposed in the previous section is privacy-preserving except for a strong adversary model where the adversary can compromise the sink and eavesdrop all network traffic. In order to deal with a compromised sink and further reduce the communication overhead, in this section we present the PASKIS protocol. This protocol differs from PASKOS in that each node selectively uses its preassigned keys in computing keyed values, and the sink does not possesses any key (being oblivious). Although PASKIS protocol has advantages over PASKOS in the above mentioned two aspects, PASKIS provides a slightly decreased privacy protection than PASKOS when a portion of sensor nodes are compromised.

5.1 Protocol description

The key idea of this protocol is that a node n_i aggregates its sensed values with its keyed values that are computed from only a subset of keys in its key ring. This subset is determined by n_i's key ring and a bitmap BIT_Q^i received from n_i's parent node along with the aggregation request.

Phase 1: *Aggregation Request Broadcast*

At the beginning, the sink connects to node n_0 and sends to n_0 the aggregation request identified by *Seed*, a number unique to each round of data aggregation. Note that n_0 becomes the root of the aggregation tree. The aggregation request message is appended with a bitmap of the keys the sink possesses, from which the *keyed values* contained in the returned aggregate will be computed by n_0. We assume that the sink does not possess any key, therefore, this bitmap is a P-bit 0-string. We denote it as BIT^0, and n_0 should enforce that in the aggregated value it sends to the sink, there is no *keyed value* computed from any key.

This process is summarized in Procedure 3.

Each node n_a, after receiving the broadcast request, including a bitmap of keys, recorded as BIT_Q^a, will relay the aggregation request to each of its children, along with a new bitmap of keys computed as follows. For each key k_i in the key pool: (1) if n_a possesses k_i, then n_a sets the i-th bit of all of its children's bitmaps to 1. This guarantees that k_i is allowed to be used in the returned aggregated result (the sum of randomized values) by n_a's children; (2) if n_a does not possess k_i but $bit_{i,Q}^a = 1$ in BIT_Q^a (it is allowed to use k_i by its parent node), then n_a will allow only one randomly chosen child to use k_i in the aggregated result; (3) if n_a does not possess k_i and it is not allowed to use k_i by its parent node, all of n_a's children will not be allowed to use k_i.

This process is summarized in Procedure 4; where $Parent(a)$ and $Children(a)$ denote the parent node and the set of children of node n_a respectively.

Phase 2: *In-network Data Aggregation*.

In the aggregation phase, each leaf node n_a computes the keyed values based on the random keys assigned to it and that $Parent(n_a)$ allowed to be used (i.e., specified by n_a in the key bitmap BIT_Q^a). Then, n_a will submit its randomized value into the aggregation along with the bitmap of the actually used keys back to its parent node (denoted as BIT_A^a) , as described in Procedure 5.

Procedure 3 *Sink_Request*

1: $BIT^0 = (0)^P$;
2: $Sink \rightarrow n_0 : \langle Seed, BIT^0 \rangle$;

Procedure 4 *Request_Relay*

1: /*Node n_a executes this procedure*/
2: Node n_a receives $\langle Seed, BIT_Q^a \rangle$ from its $Parent(n_a)$
3: **if** n_a is a leaf node **then**
4: return; //begin the aggregation procedure
5: **end if**
6: **for** each $k_i (1 \leq i \leq P)$ in the key pool **do**
7: **if** $k_i \in \mathcal{K}^a$ **then**
8: **for** each $n_j \in Children(n_a)$ **do**
9: $bit_{i,Q}^j = 1$;
10: **end for**
11: **else**
12: Random select n_k from $Children(n_a)$;
13: $bit_{i,Q}^k = bit_{i,Q}^a$;
14: **for** each $n_j \in Children(n_a) \setminus \{n_k\}$ **do**
15: $bit_{i,Q}^j = 0$;
16: **end for**
17: **end if**
18: **end for**
19: **for** each $n_j \in Children(n_a)$ **do**
20: n_a sends $< Seed, BIT_Q^j >$ to n_j.
21: **end for**

Procedure 5 *Leaf_Aggregate*

1: /*Node n_a executes this procedure*/
2: $D_a = d_a$;
3: **for** each $k_i (1 \leq i \leq P)$ in the key pool **do**
4: **if** $bit_{i,Q}^a == 1$ and $k_i \in \mathcal{K}^a$ **then**
5: $D_a = D_a + Hash(Seed, k_i)$;
6: $bit_{i,A}^a = 1$;
7: **else**
8: $bit_{i,A}^a = 0$;
9: **end if**
10: **end for**
11: n_a sends $\langle D_a, BIT_A^a \rangle$ to $Parent(n_a)$

Procedure 6 *Non-Leaf_Aggregate*

1: /*Node n_a executes this procedure*/
2: **for** each n_j in $Children(n_a)$ **do**
3: n_a receives $\langle D_j, BIT_A^j \rangle$ from n_j;
4: **end for**
5: $D_a = \sum_{n_j \in Children(n_a)} D_j$;
6: $D_a = D_a + d_a$;
7: **for** each $k_i (1 \le i \le P)$ in the key pool **do**
8: **if** $k_i \in \mathcal{K}^a$ **then**
9: $bit_{i,A}^a = bit_{i,Q}^a$;
10: $D_a = D_a + (bit_{i,A}^a - \sum_{n_j \in Children(n_a)} bit_{i,A}^j) \cdot Hash(Seed, k_i)$;
11: **else**
12: $bit_{i,A}^a = \sum_{n_j \in Children(n_a)} bit_{i,A}^j$;
13: **end if**
14: **end for**
15: **if** $n_a \neq n_0$ **then**
16: n_a sends $\langle D_a, BIT_A^a \rangle$ to $Parent(n_a)$;
17: **else**
18: n_0 sends $\langle D_0 \rangle$ to the sink;
19: **end if**

Each non-leaf node n_a first aggregates the values received from its children, then adds to that result both its own sensed value and its own keyed values. Its keyed values are computed in the following way. For each k_i in the key pool: (1) if n_a possesses k_i, it will use k_i to compute the keyed value such that the total number of times k_i is used is equal to the value $bit_{i,Q}^a$ in n_a's key bitmap BIT_Q^a; otherwise, (2) n_a will not use k_i to compute the keyed value. The final aggregation result will be sent to n_a's parent node.

This process is summarized in Procedure 6.

The last message in the aggregation process is sent by n_0 to the sink. As it will be shown in the analysis, the final aggregate D_0 will be exactly the sum of all nodes' sensed values if the original BIT_Q^0 is a P-bit 0-string, that is $D_0 = \sum_{1 \le j \le N} d_j$.

5.2 An example

We now show how the PASKIS protocol works through the same example we use for the discussion of the PASKOS protocol. The executing process is shown in Fig. 2. Figure 2(a) shows how key bitmaps propagate along with the aggregation request. Figure 2(b) shows how data aggregation proceeds along with returned key bitmaps. Eventually, the node n_0 receives all intermediate aggregates that are protected by the keyed values computed from k_1, k_4, all known to n_0. Therefore, the sink will be able to receive the sum of all sensed value from n_0.

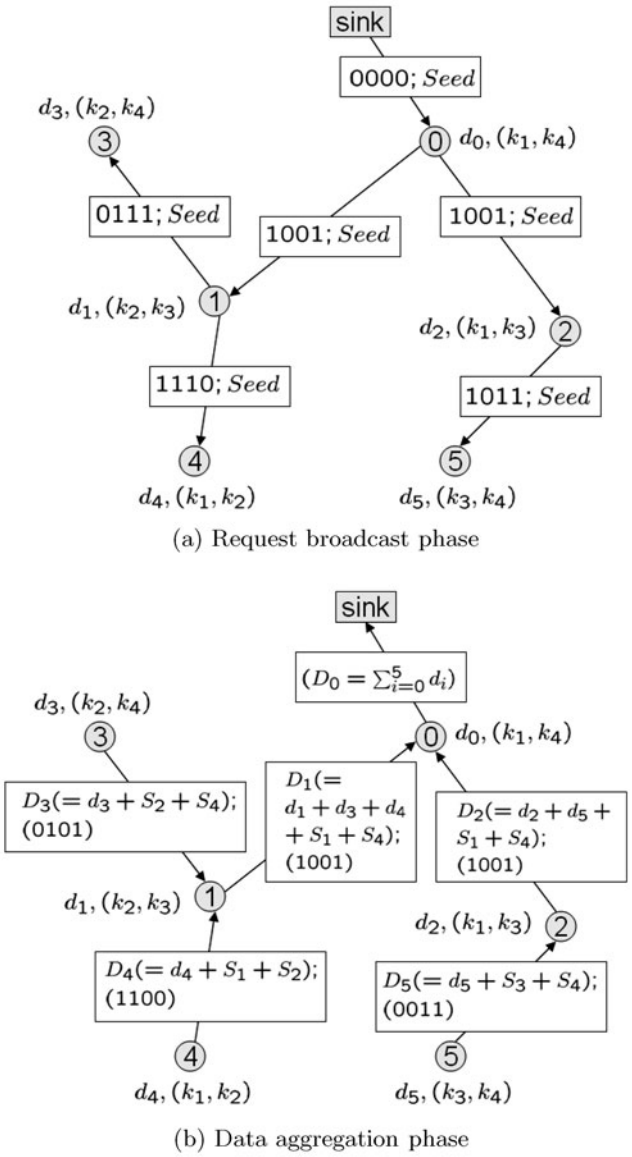


Fig. 2 In (**a**), Bitmaps (BIT_Q^i) are computed and sent by each node to its children in request broadcast phase. In (**b**) for aggregation phase, each node aggregates its sensed value with keyed values, and then send to its parent node the aggregated result along with the bitmap (BIT_A^i) of keys actually used in computing keyed values

5.3 Analysis of PASKIS protocol

Privacy The PASKIS protocol can protect the privacy of any node against any other sensor nodes in the same way as the PASKOS protocol. Note that in PASKIS, the sink does not possess any of the pre-distributed key. Therefore, it does not help the adversary if it can compromise the sink in order to obtain any node's private value. This is certainly an advantage over PASKOS protocol where the sink has the whole key pool. On the other hand, in PASKIS, fewer keys are used to compute the keyed values by each node than that of the PASKOS protocol. Therefore, if the adversary can

compromise a portion of nodes, then it can more likely learn the private sensed value of a node with PASKIS than with PASKOS. A detailed comparison of these two protocols is given in Sect. 6.

Correctness PASKIS protocol guarantees that the sink will always be able to receive exactly the aggregate of all sensed values (i.e., received data contains no keyed values at all). This is because: (1) any node introduces a new key into the computation of keyed values only if one of its ancestors possesses the same key; (2) any node removes a key from being used in the keyed values if none of its ancestors possesses that key.

Efficiency Similar to the PASKOS protocol, the PASKIS protocol also has a per-node memory and computation overhead of $O(1)$. But unlike PASKOS, the coefficient for any key in an aggregation message is limited to be either 0 or 1 in PASKIS. This is because: (1) if a node possesses a key, it will enforce the coefficient of that key in its own intermediate aggregate to be no larger than value one, as stated by the protocol; (2) if a node does not possess a key, at most only one of its children is able to use the key to compute the keyed value that is controlled by the bitmap along with the aggregation request. Therefore, the size of each aggregation message is bounded by the constant P, leading to the communication overhead $O(1)$ of the PASKIS protocol.

6 Analysis and evaluation

In this section, we present an analysis and simulation-based evaluation of our two protocols. Recall that we evaluate a privacy-preserving data aggregation protocol in terms of three criteria: effectiveness of privacy preservation, i.e., how privacy of each node is protected, (2) efficiency, i.e., how additional overhead is introduced by our protocol compare to a typical in-network aggregation protocol that does not consider the privacy preservation, and (3) and data loss resilience.

6.1 Effectiveness of privacy-preserving

We evaluate the *effectiveness of privacy-preserving* of a data aggregation protocol by examining the probability $P_{disclosure}$ that a node's private sensed value is disclosed to an adversary who compromises a portion of sensor nodes in the network and can eavesdrop all network traffic. A lower $P_{disclosure}$ indicates a more effective privacy-preserving protocol.

In the analysis of our proposal, we assume the worst case where node-to-node communications are not encrypted with pair-wise keys [32]. This allows an adversary to eavesdrop the content of the messages exchanged between the nodes. However, eavesdropping the communications between a node s and all of its neighbors is not enough to disclose the private sensed value of node s. This is because all communicated aggregates are randomized values. Thus, an adversary has to obtain the keys that are used by node s in computing keyed values. To do so, an adversary needs to compromise a number of nodes (that is not negligible [32]) in the network, or compromise the sink. In the following, we analyze our protocols in these two attack scenarios.

PASKOS protocol In this protocol, a node uses all the keys in its key ring to compute its keyed values. Therefore, the adversary has to obtain a node's whole key ring in order to infer this node's private sensed value.

Assume that the adversary: (i) has compromised C nodes other than the attack target node and (ii) it has eavesdropped all network traffic of the target node. Let $C_1, \ldots, C_K$ denote a set of events. Each C_i $(1 \le i \le K)$ represents the following event: the i-th key of the target node is not known to the adversary. Therefore, we have: $P_{disclosure} = 1 - P(C_1 \vee \cdots \vee C_K)$.

$$
\begin{aligned}
&= 1 - \Bigg(\sum_{i \in [1..K]} P(C_i) - \sum_{i_1, i_2 \in [1..k], i_1 < i_2} P(C_i \wedge C_j) \\
&\quad + \cdots + (-1)^{K-1-j} \\
&\quad \times \sum_{i_1, \ldots, i_{K-j} \in [1..K], i_1 < \cdots < i_{K-j}} P(C_{i_1} \wedge \cdots \wedge C_{i_{K-j}}) \\
&\quad + \cdots + (-1)^{K-1} P(C_1 \wedge \cdots \wedge C_K) \Bigg)
\end{aligned}
$$

The probability that i specific keys are not known by the adversary (not in any of the C nodes) is:

$$
P(C_1 \wedge \cdots \wedge C_i) = \left(\frac{\binom{P-i}{K}}{\binom{P}{K}} \right)^C
$$

where P is the key pool size, K is the key ring size (same to all nodes). Therefore, we have

$$
P_{disclosure} = \sum_{i=0}^{K} (-1)^i \binom{K}{i} \left(\frac{\binom{P-i}{K}}{\binom{P}{K}} \right)^C \tag{2}
$$

We will show later in simulations that $P_{disclosure}$ is acceptably low when the compromised nodes of the network are limited to a small portion (which might be true in practice).

Note that if the adversary is able to compromise the sink, it can obtain all the pre-distributed keys (the sink holds the entire key pool). Then, the adversary can infer any node's private sensed value.

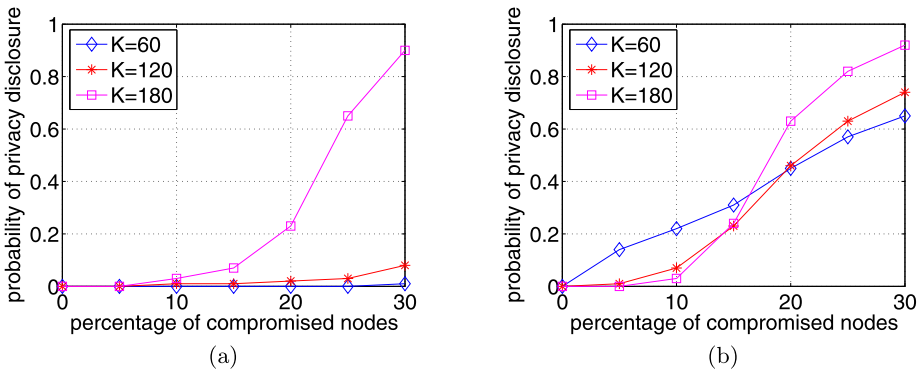


Fig. 3 (**a**) and (**b**) show the performance of PASKOS and PASKIS protocols respectively, for various key ring sizes

PASKIS protocol A sensor node selects keys from its key ring that are shared with its ancestors or descendants. A larger number of *actually* used keys by a node implies higher difficulty for an adversary to learn the privacy of this node. Thus, the worst-case privacy-disclosure scenario occurs in the node with the last number of ancestors and descendants, i.e. the leaf nodes. Suppose that a leaf node has H ancestors where H is the height of the aggregation tree. Note that H is dependent on the number of nodes, the density, and the connectivity of the network). Then, the expected number of keys that this leaf node shares with its ancestors is given by:

$$m_H = \sum_{j=0}^{K}(K-j)\left(\sum_{i=j}^{K}(-1)^{(i-j)}\binom{K}{i}\left(\frac{\binom{P-i}{K}}{\binom{P}{K}}\right)^H\right) \quad (3)$$

The probability of this leaf node's privacy is discovered by the adversary can be computed by:

$$p_H = \sum_{j=0}^{K}\left(\sum_{i=0}^{K-j}(-1)^i\binom{K}{i}\left(\frac{\binom{P-i}{K}}{\binom{P}{K}}\right)^C\right) \times \left(\sum_{i=j}^{K}(-1)^{(i-j)}\binom{K}{i}\left(\frac{\binom{P-i}{K}}{\binom{P}{K}}\right)^H\right) \quad (4)$$

where C is the number of compromised nodes. Thus, the worst-case privacy violation is given by $P_{disclosure} = p_H$.

In the PASKIS protocol, as the sink does not store any key, the adversary would not gain any advantage in compromising the sink.

Simulations To further evaluate the privacy-preserving efficacy of our protocols, we conduct simulations of a wireless sensor network where there are 200 sensor nodes distributed uniformly at random in an 1,000 m × 1,000 m area. The communication range of each sensor node is 150 m. The key pool size is $P = 2{,}000$. In all the simulation plots given in this section, each data point is the average of 1,000 runs on different random network topologies.

Our simulation results in Fig. 3(a) show that for the PASKOS protocol, decreasing the key ring size K can reduce the privacy disclosure probability. Intuitively, this is because the percentage of shared keys in any two key rings is smaller when the key ring size is smaller. This is consistent with our analytical result in (2). However, it is interesting to note that for PASKIS protocol, this trend is only true when the percentage of compromised nodes is large, as shown in Fig. 3(b). To understand this, note that a node can only use keys shared with its ancestors or descendants in the PASKIS protocol. Thus increasing the key ring size has two effects: (1) more keys potentially known to the adversary, which implies that $P_{disclosure}$ can increase; (2) more keys used by a node to compute keyed values, which implies that $P_{disclosure}$ can decrease. Figure 3(b) shows that when a large number of nodes are compromised (higher than about 15%), the first effect dominates, whereas when less than 15% nodes are compromised, the second effect dominates. These two effects can be verified by checking (3) and (4).

Comparing PASKOS vs. PASKIS We mentioned that PASKIS can provide privacy protection even when an adversary can compromise the sink, but PASKOS cannot provide privacy under this attack. However, if an adversary compromises a portion of network nodes but not the sink, then PASKOS can provide better privacy-preservation than PASKIS because nodes in PASKOS use all of their possessed keys for computing keyed values, which means that the probability for an adversary to obtain all those keys is smaller with the PASKOS protocol than with PASKIS. Comparison of the two protocols, based on the simulations for a network with the same conditions set before (also the same key pool and key ring size), is shown in Fig. 4. We see that PASKOS provides much better privacy protection than PASKIS when there is a large (that is, greater than 10%) percent of nodes are compromised.

6.2 Communication and computation overhead

Compared with a typical data aggregation protocol such as TAG [4], our protocols do not introduce additional messages, but instead append the added privacy-preserving information to regular aggregation messages. In PASKIS protocol, each regular message is added with a P-bit bitmap of

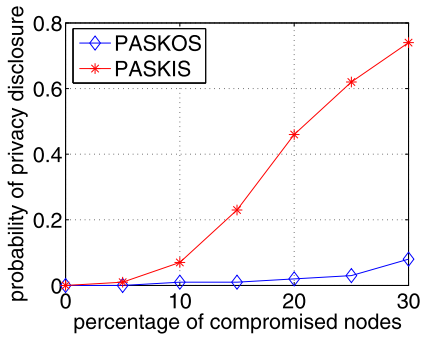


Fig. 4 Privacy-preserving effectiveness comparison between PASKOS and PASKIS

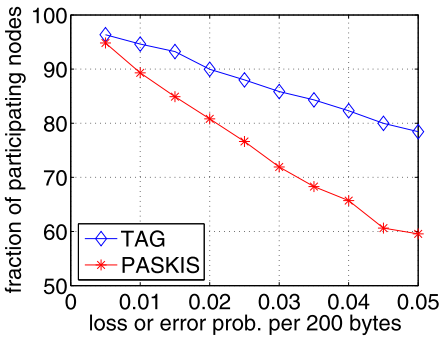


Fig. 6 Percentage of nodes actually participating in data aggregation. Comparison between TAG and PASKIS

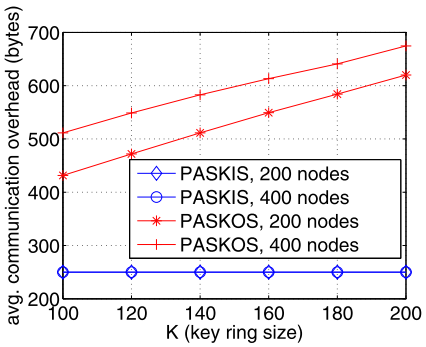


Fig. 5 Communication overhead comparison between PASKOS and PASKIS

the pre-distributed keys. Therefore, the size of each message is increased by $P/8$ bytes. In the PASKOS protocol, in the worst case, the size of a message is increased by $(\log_2 N + 1)P/8$ bytes, which happens when the height of the aggregation tree is only 1 and all the leaf-nodes choose the same coefficient for the same key. Note that, P is a pre-determined constant parameter of the privacy protection, therefore, the per-node communication complexities of PASKOS and PASKIS are $O(\log N)$ and $O(1)$, respectively. Clearly, in both proposed protocols, each node computes the keyed value using its own pre-assigned keys. The number of keys assigned to each node is k, which is smaller than P. Therefore, the per-node computation complexities of the two protocol are both $O(1)$.

To evaluate the communication overhead, we conduct simulations in the previous network, and we fix $P = 2{,}000$. Figure 5 shows the additional transferred bytes (per node) introduced by our protocols, compared with TAG. Consistent with our analysis in Sects. 4.3 and 5.3, we see that PASKIS's overhead remains a constant while PASKOS always has higher communication overhead.

6.3 Data loss resilience

Due to data loss, some readings could be lost. Hence, an aggregation result should be computed from only a portion of the nodes in the network. Solutions that enjoy this property are referred to as data loss resilient [6].

Compared with TAG [4], our protocols do not introduce any additional messages between nodes to guarantee data loss resilience. However, they increase the size of each message. Therefore, if the size-increased message of our protocols can still be encapsulated in a data packet and the packet loss rate remains the same, our protocols will have the same percentage of participating nodes as that of TAG protocol. In practice, message loss probability can be affected by not only the message size, but also by other complicating factors such as network density, error correction schemes, traffic amount, etc. In the following, we give a simple analysis, and leave a comprehensive analysis considering all those factors as our future work. First, we observe that all these factors can be summarized or abstracted as a fundamental deciding factor: per-bit loss/error probability. Assume that if a message contains an error bit, then the whole message will be discarded. If we assume a fixed loss/error probability per bit, then we can compare TAG and our PASKIS protocol in terms of the actual number of nodes participating in the data aggregation. In Fig. 6, we plot the results of a set of simulations on the same network used in previous simulations. We see that when the bit-wise loss probability is small, compared with TAG (without any privacy-protection), the network adopting our protocol (with effective privacy-protection) has a comparable percentage of nodes actually participating in the aggregation. Note in Fig. 6, x-axis is given in loss probability per 200 bytes. This is just for interpretation convenience, as TAG's message size is roughly 200 bytes [1].

Furthermore, our protocols are data-loss resilient. This is because both PASKOS and PASKIS protocols satisfy the following properties: (i) intermediate aggregation result will always be transferred as a whole, i.e., if a message is lost, corresponding nodes can be regarded as being dropped off from the aggregation without affecting other nodes; and (ii) our protocols guarantee that in retrieving the final aggregate of sensed values, a keyed value is removed only if it exists in

the received aggregated data. Therefore, the sink will get the precise aggregation result of those nodes that participated in the aggregation process.

From Table 1 we can see that many existing solutions are not data-loss resilient, including SMART [1] and the protocol in [6]. Note that the scheme proposed in [33] is not data-loss resilient unless the list of participating nodes is also received by the sink (introducing additional overhead).

7 Conclusions

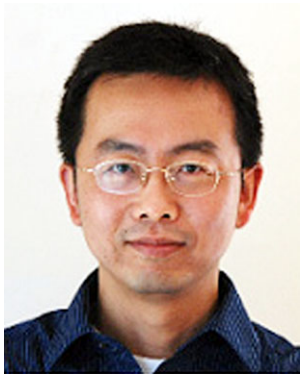
In this paper we address a relevant issue in WSNs: privacy in data aggregation. In particular, we propose two protocols, PASKOS and PASKIS, for efficient privacy-preserving data aggregation in wireless sensor networks. These two protocols provide an effective protection against both an external eavesdropper and internal threats. In particular, PASKOS can provide a strong privacy-protection for a node against any other node, or when a portion of network nodes are compromised by an adversary. PASKIS achieves privacy protection even against an adversary that can compromise the sink—trading off this feature with a slight decrease in the assured level of privacy.

To the best of our knowledge, both protocols have superior performance than other existing schemes in terms of privacy preservation and efficiency. Thorough analysis and simulation results support our findings.

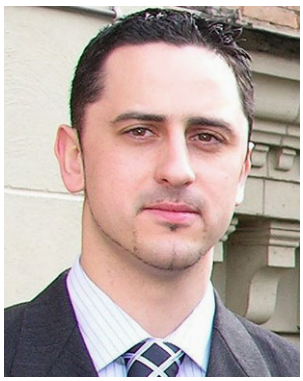
References

1. He, W., Liu, X., Nguyen, H., Nahrstedt, K., & Abdelzaher, T. (2007). PDA: Privacy-preserving data aggregation in wireless sensor networks. In *INFOCOM'07* (pp. 2045–2053).
2. Stankovic, J. A., Cao, Q., Doan, T., Fang, L., He, Z., Kiran, R., Lin, S., Son, S., Stoleru, R., & Wood, A. (2005). Wireless sensor networks for in-home healthcare: Potential and challenges. In *HCMDSS'05 workshop*.
3. De Cristofaro, E., Jarecki, S., Kim, J., & Tsudik, G. (2009). Privacy-preserving policy-based information transfer. In *Privacy enhancing technologies* (pp. 164–184).
4. Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2002). TAG: a tiny aggregation service for ad-hoc sensor networks. In *OSDI'02* (pp. 131–146).
5. Fung, W. F., Sun, D., & Gehrke, J. (2002). Cougar: the network is the database. In *SIGMOD'02* (pp. 621–621).
6. Castelluccia, C., Chan, A. C.-F., Mykletun, E., & Tsudik, G. (2009). Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN) 5*(3).
7. Girao, J., Westhoff, D., & Schneider, M. (2005). CDA concealed data aggregation for reverse multicast traffic in wireless sensor networks. In *ICC'05* (pp. 3044–3049).
8. Mlaih, E., & Aly, S. (2008). Secure hop-by-hop aggregation of end-to-end concealed data in wireless sensor networks. In *INFOCOM'08*, April 2008 (pp. 1–6).
9. Agrawal, R., & Srikant, R. (2000). Privacy-preserving data mining. In *SIGMOD'00* (pp. 439–450).
10. Evfimievski, A., Srikant, R., Agrawal, R., & Gehrke, J. (2002). Privacy preserving mining of association rules. In *KDD'02* (pp. 217–228).
11. Eschenauer, L., & Gligor, V. D. (2002). A key-management scheme for distributed sensor networks. In *CCS'02* (pp. 41–47).
12. Pietro, R. D., Mancini, L. V., Mei, A., Panconesi, A., & Radhakrishnan, J. (2008). Redoubtable sensor networks. *ACM Transactions on Information and System Security 11*(3).
13. Yang, Y., Wang, X., Zhu, S., & Cao, G. (2006). SDAP: a secure hop-by-hop data aggregation protocol for sensor networks. In *MobiHoc'06* (pp. 356–367).
14. Di Pietro, R., Michiardi, P., & Molva, R. (2009). Confidentiality and integrity for data aggregation in WSN using peer monitoring. *Security and Communication Networks*, *2*(2), 181–194.
15. Roy, S., Setia, S., & Jajodia, S. (2006). Attack-resilient hierarchical data aggregation in sensor networks. In *SASN'06* (pp. 71–82).
16. Chan, H., Perrig, A., & Song, D. (2006). Secure hierarchical in-network aggregation in sensor networks. In *CCS'06* (pp. 278–287).
17. Roy, S., Conti, M., Setia, S., & Jajodia, S. (2009). Secure median computation in wireless sensor networks. *Ad Hoc Networks (Elsevier)*, *7*(8), 1448–1462.
18. Deligiannakis, A., Stoumpos, V., Kotidis, Y., Vassalos, V., & Delis, A. (2008). Outlier-aware data aggregation in sensor networks. In *ICDE'08* (pp. 1448–1450).
19. Wagner, D. (2004). Resilient aggregation in sensor networks. In *SASN'04* (pp. 78–87).
20. Di Pietro, R., Michiardi, P., & Molva, R. (2009). Confidentiality and integrity for data aggregation in wsn using peer monitoring. *Security and Communication Networks*, *2*(2), 181–194.
21. Yao, A. (1982). Protocols for secure computations. In *FOCS'82* (pp. 160–164).
22. Cramer, R., Damgard, I., & Dziembowski, S. (2000). On the complexity of verifiable secret sharing and multiparty computation. In *STOC'00* (pp. 325–334).
23. Halpern, J., & Teague, V. (2004). Rational secret sharing and multiparty computation: extended abstract. In *STOC'04* (pp. 623–632).
24. Solanas, A., & Di Pietro, R. (2008). A linear-time multivariate micro-aggregation for privacy protection in uniform very large data sets. In *MDAI'08* (pp. 203–214).
25. Di Pietro, R., & Viejo, A. (2010). Location privacy and resilience in wireless sensor networks querying. *Computer Communications*, *34*(3), 515–523.
26. Yang, Y., Shao, M., Zhu, S., Urgaonkar, B., & Cao, G. (2008). Towards event source unobservability with minimum network traffic in sensor networks. In *WiSec 2008* (pp. 77–88).
27. De Cristofaro, E., Ding, X., & Tsudik, G. (2009). Privacy-preserving querying in sensor networks. In *ICCCN'09: proceedings of the 2009 proceedings of 18th international conference on computer communications and networks* (pp. 1–6). Washington: IEEE Comput. Soc.
28. Rivest, R., Adleman, L., & Dertouzos, M. (1978). On data banks and privacy homomorphisms. *Foundations of Secure Computation* 169–179.
29. Feng, T., Wang, C., Zhang, W., & Ruan, L. (2008). Confidentiality protection for distributed sensor data aggregation. In *INFOCOM'08*, April 2008 (pp. 56–60).
30. Conti, M., Di Pietro, R., Mancini, L. V., & Spognardi, A. (2007). FastRIPP: RFID privacy preserving protocol with forward secrecy and fast resynchronization. In *IECON 07* (pp. 52–57).

31. Zhang, L., Zhang, H., Conti, M., Di Pietro, R., Jajodia, S., & Mancini, L. V. (2010). Reverse tree-based key routing: Robust data aggregation in wireless sensor networks. In *Proceedings of the third IEEE international symposium on trust, security and privacy for emerging applications (TSP 2010)* (pp. 910–915).
32. Chan, H., Perrig, A., & Song, D. (2003). Random key predistribution schemes for sensor networks. In *S&P'03* (pp. 197–213).
33. Castelluccia, C., Mykletun, E., & Tsudik, G. (2005). Efficient aggregation of encrypted data in wireless sensor networks. In *The second annual international conference on mobile and ubiquitous systems: computing, networking and services (MobiQuitous'05)* (pp. 109–117).



Honggang Zhang holds a Ph.D. in Computer Science from University of Massachusetts Amherst. He received his B.Sc. degree from the Central South University of China, and his M.Sc. degree from Tianjin University of China. He also received a M.Sc. degree from Purdue University West Lafayette, IN, USA. He is currently an assistant professor in the Math and Computer Science Department at Suffolk University in Boston, MA, USA. His research interest is on computer networks. He has received the National Science Foundation (NSF) CAREER Award 2009–2014.



Mauro Conti received in 2005 the Laurea Degree (equivalent to M.Sc.) in Computer Science from the University of Rome "La Sapienza", Italy. He received in 2009 the Ph.D. in Computer Science from the same University. In 2008, he has been visiting scholar at the Center for Secure Information Systems (CSIS) at George Mason University, Fairfax, VA, USA. From 2009 he is a Postdoctoral Researcher at the Vrije Universiteit Amsterdam, The Netherlands. His current research interest is on security and privacy for wireless resource-constrained mobile devices (sensors, RFIDs, and mobile phones).

Roberto Di Pietro is currently an Assistant Professor at the Department of Mathematics of Università di Roma Tre—Roma, Italy. He received the Ph.D. in Computer Science from the Università di Roma "La Sapienza", Italy, in 2004. In 2004 he also received from the Department of Statistics of the same University a Specialization Diploma in Operating Research and Strategic Decisions. He received the M.Sc. in Computer Science from the University of Pisa, Italy, in 1994. He has been visiting scholar at the UNESCO Chair in Data Privacy (URV), Institut EURECOM, and George Mason University-CSIS. He has published more than one hundred research papers in the following fields: security for mobile, ad-hoc, and underwater wireless networks; security for distributed systems; access control (role mining); virtualization and cloud security; computer forensics, and applied cryptography.

Sushil Jajodia is University Professor, BDM International Professor of Information Technology, and the director of Center for Secure Information Systems at the George Mason University, Fairfax, Virginia. He has authored six books, edited thirty books and conference proceedings, and published more than 350 technical papers in the refereed journals and conference proceedings. He is the founding editor-in-chief of the Journal of Computer Security and the consulting editor of the Springer International Series on Advances in Information Security. The URL for his web page is http://csis.gmu.edu/faculty/jajodia.html.

Luigi Vincenzo Mancini received the Ph.D. degree in Computer Science from the University of Newcastle upon Tyne, UK, in 1989. From 2000, he is a full professor of Computer Science at the Dipartimento di Informatica of the University of Rome "La Sapienza". Since 1994, he is a visiting research professor of the Center for Secure Information Systems, GMU, Virginia, USA. His current research interests include: computer network and information security, secure multicast communication, public key infrastructure, authentication protocols, system survivability, computer privacy, wireless network security, fault-tolerant distributed systems, and large-scale peer-to-peer systems. He published more than 90 scientific papers in international conferences and journals which include: ACM TISSEC, IEEE TKDE, IEEE TPDS, IEEE TC and IEEE TSE. He served in the program committees of several international conferences such as: ACM Conference on Computer and Communication Security, ACM Symposium on Access Control Models and Technology, Financial Cryptography, IEEE Securecomm. He has been the Principal Investigator of many national and European research projects, and he is the founder of Information and Communication Security (ICSecurity) Laboratory, see http://icsecurity.di.uniroma1.it. Currently, he is the director of the PhD studies in Computer Science, and the director of the Master degree program in Information and Network Security of the University of Rome "La Sapienza".